O’ZBEKISTON RESPUBLIKASI ALOQA,
AXBOROTLASHTIRISH VA TELEKOMUNIKATSIYA DAVLAT QO’MITASI

TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI
FARG’ONA FILIALI

”Axborot texnologiyalari” kafedrasi

”C++ da dasturlash”
fanidan

KURS ISHI

Bajardi: 612-12 Guruh talabasi
Akbarov Shermuhammad

Qabul qildi: Bazarbayev M.
Hashimov A

Farg’ona-2015

Mavzu: Eng uzun "arra" ko’rinishidagi (tishlari yuqoriga qaragan) ketma-ket keluvchi sonlar qatori uzunligi

Reja:

Kirish

C++ dasturlash tilida ishlar ekanmiz, C++ tilining tarixi haqida to’xtalib o’tish joyiz deb o’layman.

C++ dasturlash tili C tiliga asoslangan. C esa o’z navbatida B va BCPL tillaridan kelib chiqqan. BCPL 1967 yilda Martin Richards tomonidan tuzilgan va operatsion sistemalarni yozish uchun mo’ljallangan edi. Ken Thompson o’zining B tilida BCPL ning ko’p hossalarini kiritgan va B da UNIX operatsion sistemasining birinchi versiyalarini yozgan. BCPL ham, B ham tipsiz til bo’lgan. Yani o’garuvchilarning ma’lum bir tipi bo’lmagan - har bir o’zgaruvchi kompyuter hotirasida faqat bir bayt yer egallagan. O’zgaruvchini qanday sifatda ishlatish esa, yani butun sonmi, kasrli sonmi yoki harfdekmi, dasturchi vazifasi bo’lgan.

C tilini Dennis Ritchie B dan keltirib chiqardi va uni 1972 yili ilk bor Bell Laboratoriyasida, DEC PDP-11 kompyuterida qo’lladi. C o’zidan oldingi B va BCPL tillarining juda ko’p muhim tomonlarini o’z ichiga olish bilan bir qatorda o’zgaruvchilarni tiplashtirdi va bir qator boshqa yangiliklarni kiritdi. Boshlanishda C asosan UNIX sistemalarida keng tarqaldi. Hozirda operatsion sistemalarning asosiy qismi C/C++ da yozilmoqda. C mashina arxitekturasiga bog’langan tildir. Lekin yaxshi rejalashtirish orqali dasturlarni turli kompyuter platformalarida ishlaydigan qilsa bo’ladi.

1983 yilda, C tili keng tarqalganligi sababli, uni standartlash harakati boshlandi. Buning uchun Amerika Milliy Standartlar Komiteti (ANSI) qoshida X3J11 tehnik komitet tuzildi. Va 1989 yilda ushbu standart qabul qilindi. Standartni dunyo bo’yicha keng tarqatish maqsadida 1990 yilda ANSI va Dunyo Standartlar Tashkiloti (ISO) hamkorlikda C ning ANSI/ISO 9899:1990 standartini qabul qilishdi. Shu sababli C da yozilgan dasturlar kam miqdordagi o’zgarishlar yoki umuman o’zgarishlarsiz juda ko’p kompyuter platformalarida ishlaydi.

C++ 1980 yillar boshida Bjarne Stroustrup tomonidan C ga asoslangan tarzda tuzildi. C++ juda ko’p qo’shimchalarni o’z ichiga olgan, lekin eng asosiysi u ob’ektlar bilan dasturlashga imkon beradi.

C++ funksiya va ob’ektlarning juda boy kutubxonasiga ega. Ya’ni C++ da dasturlashni o’rganish ikki qismga bo’linadi. Birinchisi bu C++ ni o’zini o’rganish, ikkinchisi esa C++ ning standart kutubxonasidagi tayyor oby’ektlarni va funksiyalarni qo’llashni o’rganishdir.

II. NAZARIY QISM

2.1 Boshqaruvchi tuzilmalar

C++ tilida qurilma kiritish va chiqarish vositalari yo'q. Bu ish standart kutubxonalarda saqlanadigan funksiyalar, turlar va obyektlar yordamida amalga oshiriladi. C++ sinflari kutubxonalaridan foyda- lanilganda, iostream.h kutubxona fayli qo'llanib, unda cin klaviatu- rasidan kiritiladigan ma'lumotlarning hamda cout displeyi ekraniga chiqariladigan ma'lumotlarning standart oqimlari, shuningdek, tegishli operatsiyalar belgilangan. Bu operatsiyalar quyidagicha:

1) << — ma'lumotlarni oqimga yozish operatsiyasi;

2) >> — ma'lumotlarni oqimdan o'qish operatsiyasi. Masalan:

```
#include<iostream.h>;

cout<<"\nElementlar miqdorini kiriting:"; cin>>n;
cout dan sonlarni chiqarishda foydalanish
cout<<1001; cout<<0.12345;
Bir paytning o'zida bir nechta qiymatni chiqarish
cout<<1<<0<<0<<1;
cout<< "Mening yaxshi ko'rgan sonim teng" <<1001; cout<< "Bu" <<20<<
"yil ichida mening oylik maoshim edi" <<493.34<<endl;
```

Agar kursorni keyingi satr boshiga surish kerak bo'lsa, chiqarish oqimiga yangi satr belgisi (\n) ni joylashtirish mumkin.

```
cout<< "Bu birinchi satr\nBu ikkinchi satr";
cout<<1<<'\n'<<0<<'\n'<<0<<'\n'<<1;
```

Dasturingiz nimaga mo'ljallanganiga qarab, ehtimol, sonlarni sakkizlik yoki o'n oltilik ko'rinishda chiqarish talab qilinib qolar. Buning uchun chiqarish oqimining ichiga dec, oct, hex modifika- torlarini joylashtirish mumkin.

```
cout << "Sakkizlik: "<<oct<< 10 <<' '<< 20 << endl; cout <<"O'n
oltilik:"<<hex<<10<<' '<< 20 << endl; cout <<"O'nlik:"<<dec << 10
<<' '<< 20 << endl;
```

Shart operator

Dasturda shoxlanishlarni tashkil qilish uchun, ya'ni ayrim fak- torlarga bog'liq holda turli xatti-harakatlar bajarilishi uchun if operatori qo'llanadi.

Operator quyidagi formatga ega:

```
if (ifoda)
{operator-1;}
[else{operator-2;}]
```

if operatorining bajarilishi ifodani hisoblashdan boshlanadi. Keyin bajarilish quyidagi sxema bo'yicha amalga oshiriladi:

- agar ifoda haqiqiy bo'lsa (ya'ni noldan boshqa), bu holda operator-1 bajariladi.
- agar ifoda soxta bo'lsa (ya'ni 0 ga teng), bu holda operator-2 bajariladi.
- agar ifoda soxta bo'lsa-yu, operator-2 bo'lmasa (shart bo'lmagan konstruksiya kvadrat qavslar ichiga olingan), bu holda if dan keyin turgan operator bajariladi.

Bu misoldan yana shu narsa ma'limki, operator-1 o'rnida, xuddi operator-2 o'rnida bo'lganidek, murakkab konstruksiyalar bo'lishi mumkin.

Solib qo'yilgan if operatorlaridan foydalanishga ham yo'l qo'yiladi. if operatori if konstruksiyasi tarkibiga yoki boshqa if operatorining else konstruksiyasiga kiritilgan bo'lishi ham mumkin.

Misollar: int t=2; int b=7;

```
int r=3;
if(t>b) {
        if(b<r) {
            r=b;
        }
```

```
}
else {
        r=t;
    }
```

Ushbu dasturning bajarilishi natijasida r=2 bo'ladi.

Davriy tuzilmalarga kirish

Ko'plab tarkorlanuvchi elementlarga ega bo'lgan algoritmga mos dasturiy kodni yaratish uchun quyidagi operatorlar tomonidan yaratiladigan davriy (siklik) tuzilmalardan foydalanish kerak.

For operatori

For operatori — davr (sikl)ni tashkil qilishning eng umumiy usuli. U quyidagi formatga ega:

for (1-ifoda; 2-ifoda; 3-ifoda) {tana}

1-ifoda odatda davrni boshqarayotgan o'zgaruvchilarning dastlabki qiymatini belgilash uchun qo'llanadi. 2-ifoda davr tanasi bajarilishi mumkin bo'lgan shartni belgilab beruvchi ifoda. 3-ifoda sikl tanasi har gal bajarilganidan so'ng siklni boshqaradigan o'zgaruvchilarning o'zgarishini belgilaydi.

For operatorini bajarish sxemasi:

1. 1-ifoda hisoblanadi.
2. 2-ifoda hisoblanadi.
3. Agar 2-ifodaning qiymati 0 dan farqli (haqiqiy) bo'lsa, davr tanasi bajariladi, 3-ifoda hisoblanadi va 2-bandga o'tish amalga oshiriladi, agar 2-ifoda 0 ga teng (yolg'on) bo'lsa, u holda boshqaruv for operatoridan keyin kelgan operatorga uzatiladi.

Shunisi muhimki, shart hammavaqt davr boshida tekshirib olinadi. Bu demak, agar bajarilish sharti avvaldan soxta bo'lsa, u holda davr tanasi biron marta ham bajarilmasligi mumkin. Misol: int i, b;

```
for (i=1; i<10; i++) {
        b=i*i;
```

```
}
```

Bu misolda 1 dan 9 gacha bo'lgan sonlar kvadratlari hisoblanadi. For operatorining ayrim variantlaridan foydalanganda, davrni boshqarayotgan bir nechta o'zgaruvchilardan foydalanish mumkinligi hisobiga uning moslashuvchanligi oshadi.

Misol:

```
int top, bot;
char string[100], temp;
for (top=0, bot=100; top<bot; top++, bot )
{
        temp=string[top]; string[bot]=temp;
}
```

Belgilar satrini aks tartibda yozishni joriy qiladigan ushbu misolda davrni boshqarish uchun ikkita top, bot o'zgaruvchilari qo'llanadi. Shuni ham ta'kidlab o'tamizki, 1- va 3-ifodaning o'rniga bu yerda vergul bilan ajratilgan va ketma-ket bajariladigan bir nechta ifoda qo'llanadi.

For operatoridan foydalanishning boshqa bir varianti bu cheksiz davrdir. Bunday davrni tashkil qilish uchun bo'sh shartli ifodadan foydalanish mumkin, davrdan chiqish uchun esa odatda qo'shimcha shartdan hamda break operatoridan (biz uni keyinroq ko'rib chiqamiz) foydalaniladi.

Misol:

```
for(;;) {
        ... break; } ...
```

C tilining sintaksisiga muvofiq operator bo'sh bo'lishi mumkinligi tufayli, for operatorining tanasi ham bo'sh bo'lishi mumkin. Operatorning bunday shakli qidirishni tashkil qilishda qo'llanishi mumkin.

Misol:

```
for(i=0; t[i]<10; i++);
```

Ushbu misolda i davrining o'zgaruvchisi t massivi birinchi elementi raqamining qiymatini oladi. Bu qiymat 10 dan oshiq bo'ladi.

While operatori

While davri operatori shartdan avvalgi davr deb ataladi va quyidagi formatga ega bo'ladi:

while (ifoda) {tana};

Ifoda sifatida C tilining har qanday ifodasidan foydalanishga, tana sifatida esa har qanday, shu jumladan, bo'sh yoki tarkibli operatordan foydalanishga yo'l qo'yiladi. While operatorini bajarish quyidagicha:

1. Ifoda hisoblanadi.

2. Agar ifoda soxta bo'lsa, u holda while operatorining bajarilishi tugallanadi hamda navbatdagi keyingi operator bajariladi. Agar ifoda haqiqiy bo'lsa, u holda while operatorining tanasi bajariladi.

3. Jarayon 1-banddan boshlab qaytariladi.

Tur davrining operatori — for (1-ifoda; 2-ifoda; 3-ifoda) {tana} — while operatori bilan quyidagicha almashtirilishi mumkin:

```
1-ifoda;
while (2-ifoda)
{
        tana 3-ifoda;
}
```

Operator for ni bajarishda bo'lganidek, while operatorida ham avval shart tekshiriladi. Shuning uchun operator tanasini bajarish hammavaqt ham talab qilinmaydigan vaziyatlarda while operatoridan foydalanish qulay.

For va while operatorlari ichida lokal o'zgaruvchilarni qo'llash mumkin bo'lib, ularni e'lon qilishda bir paytning o'zida tegishli turlar belgilanishi kerak.

Do while operatori

Do while davri operatori shart belgilangandan keyingi davr operatori deb ataladi hamda davr tanasini aqalli bir marta bajarish lozim bo'lgan holatlarda qo'llanadi. Operator formati quyidagi ko'rinishga ega:

do{tana} while (i);

Do while operatorini bajarish sxemasi:

1. Davr tanasi bajariladi (u tarkibli operator bo'lishi mumkin).
2. Ifoda hisoblanadi.
3. Agar ifoda soxta bo'lsa, bu holda do while operatorining bajarilishi tugallanadi va navbatda turgan keyingi operator bajariladi. Agar ifoda haqiqiy bo'lsa, operatorning bajarilishi 1-banddan davom etadi.

While va do while operatorlari ichiga qo'yilgan bo'lishi ham mumkin. Misol:

```
int i,j,k;

i=0; j=0; k=0;
                        do {
                          i++; j——;
        while (a[k]<i) {
                k++;
        }
}
while (i<30 && j<-30);
```

Break operatori

Break operatori uni birlashtiruvchi switch, do, for, while davrlardan eng ichkarisida joylashganining bajarilishini tugallash uchun qo'l- lanadi. Break operatorining bajarilishi tugallangach, boshqaruv tugallangan davrdan keyin turgan operatorga uzatiladi. Shu yo'l bilan muddatdan oldin davrdan chiqish ta'minlanadi.

Continue operatori

Continue operatori, break operatori kabi, faqat davr operatorlari ichida qo'llanadi, biroq, undan farqli o'laroq, dasturning bajarilishi uzilgan davrdan keyin turgan operatordan emas, balki uzilgan davr boshidan davom ettiriladi.

Misol: int a,b;

```
for (a=1, b=0; a<100; b+=a, a++)
{
        if (b%2 !=0) continue;
/*juft summalarga ishlov berish*/ }
```

Ushbu misolda ko'p nuqta bilan ifodalangan xatti-harakatlar faqat toq b lardagina bajariladi, chunki 1 dan a gacha bo'lgan sonlar summasi toq bo'lib, qolgani uchun continue operatori boshqaruvni for davrining navbatdagi o'ramiga uzatadi hamda bunda ishlov berish operatorlarini bajarmaydi. Continue operatori, break operatori kabi, o'zini o'rab turgan davrlarning eng ichkaridagisini uzib qo'yadi.

Kalit bo'yicha o'tish operatori

Kalit bo'yicha o'tish — switch operatori umumiy ko'rinishi quyidagicha:

```
Switch(<ifoda>) {
Case <1-qiymat>:<1-operator>

break;

default: <operator>

case: <n-operator>; }
```

Oldin qavs ichidagi butun ifoda hisoblanadi va uning qiymati hamma variantlar bilan solishtiriladi. Biror variantga qiymat mos kelsa, shu variantda ko'rsatilgan operator bajariladi. Agar biror variant mos kelmasa, default orqali ko'rsatilgan operator bajariladi. Break operatori ishlatilmasa shartga mos kelgan variantdan tashqari keyingi variantdagi operatorlar ham avtomatik bajariladi.

Default; break va belgilangan variantlar ixtiyoriy tartibda kelishi mumkin. Default yoki break operatorlarini ishlatish shart emas. Belgilangan operatorlar bo'sh bo'lishi ham mumkin. Misol tariqasida bahoni son miqdoriga qarab aniqlash dasturini ko'ramiz:

```
#include <iostream.h> int baho; cin>>
baho;
switch(baho)
{case 2:cout <<" \n yomon";break; case 3:cout <<"
\n o'rta";break; case 4:cout <<" \n yaxshi";break;
case 5:cout <<" \n a'lo";break; default:cout <<" \n
baho noto'g'ri kiritilgan";
                                } }
```

Keyingi misolimizda kiritilgan simvol unli harf ekanligi aniqlanadi:

```
#include <iostream.h>
int baho; char c; cin >> c;
switch(c)
{case ' a ':
case ' u ':
case ' o ':
case ' i ':
cout <<"\n Kiritilgan simvol unli harf ";break;
default: cout <<" \n Kiritilgan simvol unli harf emas "; }
```

2.1 C++ dasturlash tilida funksiyalar va massivlar bilan ishlash

Dastur hajmining ko'payishi bilan uning xotirasida hamma detal- larni saqlab turish imkoni qiyinlashadi. Dasturni soddalashtirish uchun u qismlarga bo'linadi. C++ da masala funksiyalar yordamida soddaroq masalachalarga bo'linishi mumkin. Shuningdek, masalaning funksiyalarga bo'linishi kodning

ortiqchaliligini bartaraf etish imkonini ham beradi, chunki funksiya bir marta yoziladi, ko'p marta chaqiriladi. Tarkibida funksiya bo'lgan dasturni sozlash oson bo'ladi.

Ko'pincha qo'llanayotgan funksiyalarni kutubxonalarga joylash- tirish mumkin. Shunday qilib, sozlashda va kuzatib borishda ancha sodda dasturlar yaratiladi.

Funksiyalarni e'lon qilish va aniqlash

Funksiya — bu tavsiflar va operatorlarning nomlangan ketma- ketligi bo'lib, tugallangan xatti-harakatlarni, masalan, massivni shakllantirish, massivni bosib chiqarish va h.k. larni bajaradi.

Funksiya, birinchidan, C++ ning hosila turlaridan biri, ikkin- chidan esa, minimal bajarilayotgan dastur moduli hisoblanadi.

Har qanday funksiya e'lon qilinishi va aniqlanishi kerak.

Funksiyani e'lon qilishda (prototip, sarlavha) unga nom, qayta- rilayotgan qiymat turi va uzatilayotgan parametrlar ro'yxati beriladi.

Funksiyaning aniqlanishi, e'londan tashqari, yana tavsiflar va operatorlar ketma-ketligidan iborat funksiya tanasini bildiradi.

Funksiyaning turi nomi([formal_parametrlar_ro'yxati])

(funksiya tanasi)

Funksiya tanasi bu — blok yoki tarkibli operatordir. Funksiya ichida boshqa funksiyani aniqlash mumkin emas. Funksiya tanasida funksiyaning olingan qiymatini chaqirilish nuqtasiga qaytaradigan operator bo'lishi lozim. U ikkita shaklga ega bo'ladi:

1) return ifoda;
2) return.

Birinchi shakl natijani qaytarish uchun qo'llanadi, shuning uchun aniqlashdagi funksiya qanday turga ega bo'lsa, ifoda ham shunday turga ega bo'lishi kerak. Agar funksiya qiymatni qaytarmasa, ya'ni tur void ga ega bo'lsa,

ikkinchi shakl qo'llanadi. Dasturchining o'zi bu operatorni funksiya tanasida qo'llamasligi mumkin, kompilyator uni funksiya oxiriga avtomatik tarzda qo'shib qo'yadi.

Qaytarilayotgan turning qiymati, massiv va funksiyadan tashqari, har qanday turdagi qiymat bo'lishi mumkin, ammo massiv yoki funksiyaga ko'rsatkich ham bo'lishi mumkin.

Formal parametrlar ro'yxati — bu funksiyaga uzatilishi lozim bo'lgan qiymatlar. Ro'yxat elementlari vergullar bilan ajratiladi. Har bir parametr uchun tur va nom ko'rsatiladi. E'londa nomlarni ko'rsat- masa ham bo'ladi.

Funksiya tanasida yozilgan operatorlar bajarilishi uchun funksiyani chaqirib olish lozim. Chaqirishda funksiyaning nomi va faktik parametrlari ko'rsatiladi. Funksiya tanasi operatorlarini bajarishda faktik parametrlar formal parametrlarning o'rnini egallaydi. Faktik va formal parametrlar miqdori va turiga ko'ra bir-biriga mos kelishi kerak.

Kompilyator chaqirilishning to'g'riligini tekshirish imkoniga ega bo'lishi uchun funksiyani e'lon qilish funksiya chaqirilishdan oldin matnda bo'lmog'i lozim. Agar funksiya void bo'lmagan turga ega bo'lsa, u holda uning chaqirilishi ifodaning operatsiya bajarilayotgan elementi bo'lishi mumkin.

Misol:

Aytaylik, uchburchak tomonlarining koordinatalari berilgan. Agar shunday uchburchak mavjud bo'lsa, uning maydoni topilsin.

I. Matematik model:

1) I=sqtr(pow(x1-x2,2)+pow(y1-y2,2));//uchburchak tomonining uzunligi;

2) p=(a+b+c)/2;

s=sqrt(p*(p-a)*(p-b)*(p-c));//Geron formulasi;

3) uchburchak mavjudligini tekshirish (a+b>c&&a+c>b&&c+b>a)

II. Algoritm:

1) (x1,u1), (ch2,u2), (ch3,u3) uchburchagi tomonlarining koor- dinatlari kiritilsin;

2) ab, bc, ca tomonlarining uzunligi hisoblansin;

3) shunday tomonlarga ega bo'lgan uchburchakning mavjudligi tekshirilsin. Agar mavjud bo'lsa, unda uning maydoni hisoblansin va natijasi chiqarilsin;

4) agar mavjud bo'lmasa, xabar chiqarilsin;

5) agar hamma koordinatlar 0 ga teng bo'lsa, unda tamom, aks holda 1-bandga qaytiladi.

```
#include<iostream.h>
#include<math.h>
double line(double x1, double y1, double x2, double y2) {
//Funksiya x1,y1 x2, y2 koordinatalariga ega bo'lgan kesim uzunligini qaytarib beradi:
return sqrt(pow(x1-x2,2)+pow(y1-y2,2));
}
double square(double a, double b, double c); {
//funksiya a, b, c uzunlikdagi tomonlarga ega bo'lgan uchburchak maydonini qaytarib beradi. double s, r=(a+b+c)/2;
return s=sqrt(p*(p-a)*(p-b)*(p-c));//Geron formulasi }
bool triangle(double a, double b, double c);
{
//agar uchburchak mavjud bo'lsa, true ni qaytarib beradi;
if(a+b>&&a+c>b&&c+b>a)return true;
else return false; }
void main() {
double x1=1,y1,x2,y2,x3,y3; double point1_2,point1_3,point2_3;
do {
cout<<"\n Uchburchak koordinatalari:";
cin>>x1>>y1>>x2>>y2>>x3>>y3;
point1_2=line(x1,y1,x2,y2);
point1_3=line(x1,y1,x3,y3);
point2_3=line(x2,y2,x3,y3);
```

```
if(triangle(point1_2,point1_3,point2_3)==true)
cout<<"S="<<square(point1_2,point2_3,point1_3)<<'\n';
else cout<<"\n Uchburchak mavjud emas"; }
while(!(x1==0&&y1==0&&x2==0&&y2==0&&x3==0&&y3==0)); }
```

Funksiyalar prototiplari

Funksiyaga murojaat qilish mumkin bo'lsin uchun xuddi shu faylning o'zida funksiya aniqlovchisi yoki tavsifi (prototipi) bo'l- mog'i lozim.

double line(double x1, double y1, double x2 double y2); double square(double a, double b, double c); double triangle(double a, double b, double c); double line(double, double, double, double); double square(double, double, double); double triangle(double, double, double).

Bu yuqorida tavsiflari keltirilgan funksiyalarning prototiplaridir. Prototiplar bo'lganda, chaqirilayotgan funksiyalar chaqirayotgan funksiyalar bilan bitta faylda bo'lishlari shart emas, balki ular alohida modullar ko'rinishida rasmiylashtirilishi hamda ko'chirilgan holda obyektlar modullari kutubxonasida saqlanishlari mumkin. Xuddi shu narsa standart modullardagi funksiyalarga ham tegishli. Bu holda obyekt modullari sifatida translatsiya qilinib, rasmiylashtirilib bo'lingan kutubxona funksiyalarining aniqlovchilari kompilyator kutubxonasida bo'ladi, funksiyalar tavsiflarini esa dasturga qo'shimcha ravishda kiritish lozim bo'ladi. Bu ish include<fayl nomi> protsessor buyruqlari yordamida amalga oshiriladi.

Fayl_nomi sarlavhaviy faylni aniqlaydi. Sarlavhaviy fayl esa be- rilgan funksiyalar kompilyatori uchun standart bo'lgan guruhlar prototipiga ega bo'ladi. Masalan, deyarli barcha dasturlarda biz kiritish-chiqarish obyektlar oqimining tavsifi uchun #include<iostream.h> buyrug'idan hamda ularga mos operatsiyalardan foydalandik.

Katta miqdordagi funksiyalardan iborat bo'lgan hamda turli modullarda joylashtirilgan dasturlarni ishlab chiqishda funksiyalar prototiplari va tashqi obyektlarning tavsiflari (konstantalar, o'zgaruv- chilar, massivlar) alohida faylga

joylashtiriladi. Bu fayl esa include "fayl_nomi" direktivasi yordamida har bir modulning boshiga kiritiladi.

Funksiya parametrlari

Chaqirilayotgan va chaqirayotgan funksiyalar o'rtasida axborot almashinishning asosiy usuli bu parametrlardir. Parametrlarni funksiyaga uzatishning ikkita usuli mavjud: manzil bo'yicha va qiymati bo'yicha.

Qiymati bo'yicha uzatishda quyidagi xatti-harakatlar bajariladi:

a) faktik parametrlar o'rnida turgan ifodalar qiymatlari hisob- lanadi;

b) funksiyaning formal parametrlari uchun stekda xotira ajratiladi;

d) har bir faktik parametrga formal parametr qiymati beriladi,

bunda turlarning o'zaro muvofiqligi tekshiriladi hamda zarurat tug'ilganda ular qayta o'zgartiriladi.

Misol:

```
double square(double a, double b, double c);
{
//funksiya a, b, c uzunlikdagi tomonlarga ega bo'lgan uchburchak maydonini qaytarib beradi.
double s, r=(a+b+c)/2;
return s=sqrt(p*(p-a)*(p-b)*(p-c));//Geron formulasi
}
```

Shunday qilib, stekka faktik parametrlarning nusxalari kiritiladi va funksiya operatorlari ushbu nusxalar bilan ish olib boradi. Faktik parametrlarning o'ziga funksiyaning kirish huquqi yo'q, demak, ularni o'zgartirish imkoni ham yo'q.

Manzil bo'yicha uzatishda stekka parametrlar manzillarining nusxalari kiritiladi, demakki, funksiyada faktik parametr joylash- tirilgan xotira uyasiga kirish huquqi paydo bo'ladi va funksiya bu parametrni o'zgartirishi mumkin.

```
void change(int*a,int*b)//manzil bo'yicha uzatish;
{
```

```
int r=*a; *a=*b;*b=r;}
int x=1,y=5;
change(&x,&y);
cout<<"x="<<x<<"y="<<y;
x=5y=1 kelib chiqadi.
```

Manzil bo'yicha uzatish uchun iqtiboslar ham qo'llanishi mumkin. Iqtibos bo'yicha uzatishda funksiyaga chaqirish paytida ko'rsatilgan parametr manzili uzatiladi, funksiya ichida esa parametrga barcha murojaatlarning sezilmagan holda nomlari bekor qilinadi:

```
void change(int &a,int &b) {int r=a; a=b;b=r;}
int x=1,y=5; change(x,y);
cout<<"x="<<x<<"y="<<y; x=5y=1 kelib
chiqadi.
```

Ko'rsatkichlar o'rniga iqtiboslardan foydalanish dasturning o'qilishini yaxshilaydi, chunki bu o'rinda nomni bekor qilish operatsiyasini qo'llamasa ham bo'ladi. Qiymat bo'yicha uzatish o'rniga iqtiboslardan foydalanish samaraliroq hamdir, chunki parametrlarni nusxalashni talab qilmaydi. Agar funksiya ichida parametrning o'zgarishini taqiqlash lozim bo'lib qolsa, bu holda const modifikatori qo'llanadi. Const modifikatorini funksiyada o'zgarishi ko'zda tutilmagan barcha parametrlar oldidan qo'yish tavsiya qilinadi (undagi qaysi parametrlar o'zgaradi-yu, qaysilari o'zgarmasligi sarlavhadan ko'rinib turadi).

Lokal va global o'zgaruvchilar

Berilgan funksiya ichida qo'llanadigan o'zgaruvchilar lokal deb ataladi. Ular uchun stekda xotira ajratilmaydi, shuning uchun, ish tugagach, funksiyalar xotiradan chiqarib tashlanmaydi. Ko'rsatkichni lokal o'zgaruvchiga qaytarish mumkin emas, chunki bunday o'zgaruvchi ajratib bergan xotira bo'shatila boshlaydi:

```
int*f() {
int a;

return&a;//NOTO'G'RI }
```

Global o'zgaruvchilar — bu funksiyadan tashqarida tavsiflangan funksiyalar. Ular shunday nomli lokal funksiyalar bo'lmagan barcha funksiyalarda ko'rinadi.

Misol:

```
int a,b;//global o'zgaruvchilar
void change() {
int r;//lokal o'zgaruvchi
r=a;a=b;b=r; }
void main() {
cin>>a>>b; change();
cout<<"a="<<a<<"b="<<b; }
```

Funksiyalar o'rtasida ma'lumotlarni uzatish uchun global o'zgaruvchilardan ham foydalanish mumkin, lekin bunday qilish tavsiya etilmaydi, chunki bu dasturni sozlashni qiyinlashtiradi hamda funksiyalarni kutubxonaga joylashga to'sqinlik qiladi. Funksiyalar maksimal mustaqil bo'lishiga, funksiya prototipi esa ularning interfeysini to'lig'icha aniqlashiga intilish kerak.

Dastlabki (yashirilgan) parametrlar qiymatiga ega bo'lgan funksiyalar

Funksiyani aniqlashda dastlabki (yashirilgan) parametr qiymati bo'lishi mumkin. Agar funksiyani chaqirishda tegishli parametr tushirib qoldirilgan bo'lsa, mana shu qiymat qo'llanadi. Bunday parametrning o'ng tomonida tavsiflangan parametrlar ham yashirilgan bo'lishi lozim.

Misol:

```
void print(int value=1)
{

cout<< '\n'<< "Uy raqami: "<<value:}
```

Chaqirishlar:

1. print();

   Xulosa: Uy raqami: 1

2. print(15);

   Xulosa: Uy raqami: 15

Taqdim etiladigan (inline) funksiyalar

C++ dagi ayrim funksiyalarni inline rasmiy so'zini qo'llagan holda aniqlash mumkin. Bunday funksiya taqdim etilayotgan yoki o'rnatilayotgan funksiya deb ataladi.

Masalan:

inline float line(float x1, float y1, float x2=0, float y2=0)

{return sqrt(pow(x1-x2)+pow(y1-y2,2));}// funksiya (x1,u1) koordinatali nuqtadan (x2,u2) koordinatali nuqtagacha bo'lgan masofani orqaga qaytaradi.

Qo'yilgan funksiyaning har bir chaqirishiga ishlov berar ekan, kompilyator dastur matniga dastur tanasi operatorlari kodini joy- lashtirishga urinadi. Inline spetsifikatori funksiya uchun ichki bog'lash- ni aniqlaydi. Ichki bog'lash shundan iboratki, bunda kompilyator funksiyani chaqirish o'rniga funksiya kodining buyruqlarini qo'yadi. Bunda dastur hajmi kattalashishi mumkin, ammo chaqirilayotgan funksiya boshqaruvni uzatish va undan qaytishga ketadigan sarflar bo'lmaydi. Agar funksiya tanasi bir necha operatorlardan iborat bo'lsa, o'rniga o'rin qo'yiladigan funksiyalar qo'llanadi.

Quyidagi funksiyalar o'rniga o'rin qo'yiladigan bo'la olmaydi:

1. Rekursiv funksiyalar.
2. Chaqirilishi funksiyalarning aniqlanishidan oldin joylash- tiriladigan funksiyalar.
3. Ifodada bittadan ortiq marta chaqiriladigan funksiyalar.
4. Davrlar, ulagichlar va uzatish diapazonlariga ega bo'lgan funksiyalar.
5. O'rniga o'rin qo'yishni amalga oshirish uchun o'ta katta hajmga ega bo'lgan funksiyalar.

Funksiyalarni ortiqcha yuklash

Ortiqcha yuklashning maqsadi shundan iboratki, bunda bitta nomga ega bo'lgan funksiya turlicha bajarilishi kerak hamda unga murojaat qilinganda har xil turlarga va har xil sondagi faktik parametrlarga ega bo'lgan har xil qiymatlarni qaytarib berishi kerak. Ortiqcha yuklashni ta'minlash uchun har bir ortiqcha yuklangan funksiya uchun qaytarib berilayotgan qiymatlar va uzatilayotgan parametrlarni aniqlash kerak. Bu ish shunday amalga oshirilishi kerakki, bunda har bir ortiqcha yuklangan funksiya xuddi shu nomli boshqa funksiyadan ajralib tursin. Kompilyator faktik parametrlar turi bo'yicha qanday funksiyani tanlab olishni aniqlab beradi.

Misol:

```
#include<iostream.h>
#include<string.h>
int max(int a, int b) {
if(a>b) return a;
else return b; }
float max(float a, float b) {
if(a>b) return a;
else return b; }
void main() {
int a1,b1;
float a2, b2;
cout<< "\nfor int:\n";
cout<<"a=?"; cin>>a1;
cout<<"b=?"; cin>>b1;
cout<< "\nMAX="<<max(a1,b1)<< '\n';
cout<<"\nfor float:\n";
cout<<"a=?"; cin>>a2;
cout<<"b=?"; cin>>b2;
cout<< "\nMAX="<<max(a2,b2)<< '\n';
```

Ortiqcha yuklangan funksiyalarni tavsiflash qoidalari:

1) Ortiqcha yuklangan funksiyalar bitta ko'rish sohasida joylashtirilgan bo'lishi kerak.

2) Ortiqcha yuklangan funksiyalar yashiringan parametrlarga ega bo'lishi mumkin, bunda turli funksiyalardagi bitta parametrning qiymatlari o'zaro mos bo'lishi kerak. Ortiqcha yuklangan funksiya- larning turli variantlarida turli miqdordagi yashiringan parametrlar bo'lishi mumkin.

3) Agar funksiyalar parametrlarining tavsifi faqat const modifi- katori bilan yoki iqtibosning mavjudligi bilan farqlansa, funksiyalar ortiqcha yuklangan bo'lolmaydi.

Masalan, int&f1(int&,const int&){...} va int f1(int,int){...} funksiyalari ortiqcha yuklangan emas, chunki funksiyalarning qaysi biri chaqirila- yotganini kompilyator bila olmaydi: parametrni qiymat bo'yicha uza- tayotgan hamda parametrni manzil bo'yicha uzatayotgan funksiyalarning chaqirilishi o'rtasida sintaktik (ma'no bo'yicha) farq yo'q.

Massivlar bilan ishlash

Massiv tushunchasi.

Massiv — bu bitta turga mansub bir nechta o'zgaruvchilar to'plami. TYPE turidagi LENGTH ta elementdan iborat a nomli massiv shunday e'lon qilinadi: type a[length];

Bu maxsus a[0], a[1], ..., a[length-1] nomlarga ega bo'lgan type turidagi o'zgaruvchilarning e'lon qilinishiga to'g'ri keladi. Massivning har bir elementi o'z raqamiga — indeksiga ega. Massivning x- elementiga kirish indekslash operatsiyasi yordamida amalga oshiriladi: int x=...; //butun sonli indeks

TYPE value=a[x]; //x-elementni o'qish a[x]=value; //x-elementga yozish

Indeks sifatida butun tur qiymatini chiqarib beradigan har qanday ifoda qo'llanishi mumkin: char, short, int, long. C da massiv elementlarining

indekslari 0 dan boshlanadi (1 dan emas), LENGTH elementdan iborat bo'lgan massivning oxirgi elementining indeksi esa — bu LENGTH-1 (LENGTH emas). Shuning uchun massivning barcha elementlari bo'yicha davr — bu: TYPE a[LENGTH]; int indx; for(indx< LENGTH; indx++) ...a[indx]...;

indx< LENGTH ning qiymati indx<= LENGTH-1 qiymatiga teng. Massiv chegarasidan tashqariga chiqish (ya'ni mavjud bo'lmagan elementni o'qish-yozishga urinish) dastur xulq-atvorida kutilmagan natijalarga olib kelishi mumkin. Shuni ta'kidlab o'tish joizki, bu eng ko'p tarqalgan xatolardan biridir.

Statik massivlarni nomlab e'lon qilish mumkin, bunda massivlar elementlarining qiymatlari vergul bilan ajratilgan shakldor qavs {} ichida sanab o'tiladi. Agar massiv uzunligiga qaraganda kamroq element berilgan bo'lsa, qolgan elementlar 0 hisoblanadi:

```
int a10[10]={1, 2, 3, 4}; //va 6 ta nol
```

Agar nomlangan massivning tavsifida uning o'lchamlari ko'rsa- tilmagan bo'lsa, u kompilyator tomonidan sanab chiqiladi:

```
int a3[]={1, 2, 3};          //go'yo a3[3]
```

Massivlarni navlarga ajratish

Navlarga ajratish — bu berilgan ko'plab obyektlarni biror bir belgilangan tartibda qaytadan guruhlash jarayoni.

Massivlarning navlarga ajratilishi tez harakatlanuvchiligiga ko'ra farqlanadi. Navlarga ajratishning n*n ta qiyoslashni talab qilgan oddiy usuli va n*In(n) ta qiyoslashni talab qilgan tez usuli mavjud. Oddiy usullar navlarga ajratish tamoyillarini tushuntirishda qulay hisoblanadi, chunki sodda va kalta algoritmlarga ega. Murakkablashtirilgan usullar kamroq sonli operatsiyalarni talab qiladi, biroq operatsiyalarning o'zi murakkabroq, shu sababli uncha katta bo'lmagan massivlar uchun oddiy usullar ko'proq samara beradi.

Oddiy usullar uchta asosiy kategoriyaga bo'linadi:

— oddiy kiritish usuli bilan navlarga ajratish;

— oddiy tanlash usuli bilan navlarga ajratish;

— oddiy almashtirish usuli bilan navlarga ajratish.

Oddiy kiritish usuli bilan navlarga ajratish

Massiv elementlari avvaldan tayyor berilgan va dastlabki ketma- ketliklarga bo'linadi. 1=2 dan boshlab, har bir qadamda dastlabki ketma- ketlikdan 1-element chiqarib olinadi hamda tayyor ketma-ketlikning kerakli o'rniga kiritib qo'yiladi. Keyin bittaga ko'payadi va h.k.

44	**55**	**12**	**42**	**94**	**18**

Tayyor dastlabki ketma-ketlik

Kerakli joyni izlash jarayonida ko'proq o'ndan bitta pozitsiyadan tanlab olingan elementni uzatish amalga oshiriladi, ya'ni tanlab olingan element, J:=I-1 dan boshlab, navlarga ajratib bo'lingan qismning navbatdagi elementi bilan qiyoslanadi. Agar tanlab olingan element a[I] dan katta bo'lsa, uni navlarga ajratish qismiga qo'shadilar, aks holda a[J] bitta pozitsiyaga suriladi, tanlab olingan elementni esa navlarga ajratilgan ketma-ketlikning navbatdagi elementi bilan qiyoslaydilar. To'g'ri keladigan joyni qidirish jarayoni ikkita turlicha shart bilan tugallanadi:

— agar a[j]>a[i] elementi topilgan bo'lsa;

— agar tayyor ketma-ketlikning chap uchiga yetilgan bo'lsa.

```
int i, j, x;
fjr(i=1; i<n; i++)
{
x=[i];// kiritib qo'yishimiz lozim bo'lgan elementni esda saqlab qolamiz
j=i-1;
while(x<a[j]&&j>=0)//to'g'ri keladigan joyni qidirish
}
```

```
a[j+1]=a[j];//o'ngga surilish
j——;
}
a[j+1]=x;//elementni kiritish
}
```

Oddiy tanlash usuli bilan navlarga ajratish

Massivning minimal elementi tanlanadi hamda massivning birinchi elementi bilan joy almashtiriladi. Keyin jarayon qolgan elementlar bilan takrorlanadi va h.k.

44	**55**	**12**	**42**	**94**	**18**

```
int i,min,n, n_ min,j;
for(i=0;i<n—1;i++) {
min=a[i];n_min=i; //minimal qiymatni qidirish
for(j=i+1;j<n;j++)
        if(a[j]<min){min=a[j];n+min=j;}
        a[n_min]=a[i];//almashtirish a[i]=min;
}
```

Oddiy almashtirish usuli bilan navlarga ajratish

Elementlar juftlari oxirgisidan boshlab qiyoslanadi va o'rin almashinadi. Natijada massivning eng kichik elementi uning eng chapki elementiga aylanadi. Jarayon massivning qolgan elementlari bilan davom ettiriladi.

44	**55**	**12**	**42**	**94**	**18**

```
for(int i=1;i<n;i++)
for(int j=n—1;j>=i;j )
if(a[j]<a[j—1])
{int r=a[j];a[j]=a[j—1];a[j—1]=r;}
```

}

Ko'p o'lchamli massivlar

C++ da massivning eng umumiy tushunchasi — bu ko'rsatkichdir, bunda har xil turdagi ko'rsatkich bo'lishi mumkin, ya'ni massiv har qanday turdagi elementlarga, shu jumladan, massiv bo'lishi mumkin bo'lgan ko'rsatkichlarga ham ega bo'lishi mumkin. O'z tarkibida boshqa massivlarga ham ega bo'lgan massiv ko'p o'lchamli hisoblanadi.

Bunday massivlarni e'lon qilishda kompyuter xotirasida bir nechta turli xildagi obyekt yaratiladi. Masalan, int arr[4][3]: Arr ***/***

arr[0]	arr[0][0]	arr[0][1]	arr[0][2]
arr[1]	arr[1][0]	arr[1][1]	arr[1][2]
arr[2]	arr[2][0]	arr[2][1]	arr[2][2]
arr[3]	arr[3][0]	arr[3][1]	arr[3][2]

Shunday qilib, arr[4][3] ning e'lon qilinishi dasturda uchta turli xildagi obyektlarni yuzaga keltiradi: arr identifikatorli ko'rsatkichni, to'rtta ko'rsatkichdan iborat nomsiz massivni va int turidagi o'n ikkita sondan iborat nomsiz massivni. Nomsiz massivlarga kirish huquqiga ega bo'lish uchun arr ko'rsatkichli adresli ifodalar qo'llanadi. Ko'rsatkichlar massivi elementlariga kirish huquqi arr[2] yoki *(arr+2) shaklidagi indeksli ifodaning bittasini ko'rsatish orqali amalga oshiriladi. Int turidagi ikki o'lchamli sonlar massiviga kirish uchun arr[1][2] shaklidagi ikkita indeksli ifoda yoki unga ekviva- lent bo'lgan *(*(arr+1)+2) va (*(arr+1))[2] shaklidagi ifodalar qo'llanishi kerak. Shuni ham hisobga olish kerakki, C tili sintaksisi nuqtayi nazaridan arr ko'rsatkichi va arr[0], arr[1], arr[2], arr[3] ko'rsatkichlari konstantalardir hamda ularning qiymatlarini dasturni bajarish paytida o'zgartirish mumkin emas. Uch o'lchamli massivni joylashtirish ham xuddi shunga o'xshash amalga oshiriladi hamda float arr3[3][4][5] ning e'lon qilinishi dasturda, float turidagi oltmishta sondan iborat uch o'lchamli massivning o'zidan tashqari, float turiga tuzilgan to'rtta ko'rsatkichdan iborat massivni, float ko'rsatkichlar massiviga tuzilgan uchta ko'rsatkichdan iborat massivni va float ga tuzilgan ko'rsatkichlar

massivining massivlariga ko'rsat- kichni yuzaga keltiradi. Ko'p o'lchamli massivlar elementlarini joylashtirishda ular xotirada satrlar bo'yicha bir tartibda joylash- tiriladi, ya'ni oxirgi indeks hammadan tezroq o'zgaradi, birinchisi esa sekinroq o'zgaradi. Bunday tartib ko'p o'lchamli massiv boshlang'ich elementining adresini hamda faqat bitta indeks ifodasini qo'llab, ko'p o'lchamli massivning har qanday elementiga murojaat qilish imkonini beradi.

Masalan, arr[1][2] elementiga murojaatni ptr2 ko'rsatkichi yordamida amalga oshirsa bo'ladi. Bu ko'rsatkich esa ptr2[1*4+2] () murojaati yoki ptr2[6] murojaati sifatida int *ptr2=arr[0] shaklida e'lon qilingan bo'ladi. Ta'kidlab o'tish lozimki, tashqi tomondan o'xshash arr[6] murojaatini bajarish mumkin emas, chunki 6 indeksli ko'rsatkich mavjud emas.

Shuningdek, uch o'lchamli massivga kiradigan arr3[2][3][4] elementiga murojaat uchun float *ptr3=arr3[0][0] ko'rinishida tavsiflangan, ptr3[3*2+4*3+4] yoki ptr3[22] shaklidagi bitta indeksli ifodaga ega bo'lgan ko'rsatkichni qo'llash mumkin.

Ko'rsatkichlar massivlari

Ko'rsatkichlar massivlari quyidagicha ta'riflanadi:

<tip> *<nom>[<son>]

Misol uchun int *pt[6] ta'rif int tipidagi obyektlarga olti ele- mentli massivni kiritadi. Ko'rsatkichlar massivlari satrlar massivlarini tasvirlash uchun qulaydir. Misol uchun familiyalar ro'yxatini kiritish uchun ikki o'lchovli massivdan foydalanish kerak: char fam[][20]={ "Olimov", "Rahimov", "Ergashev"}

Bunday po'yxat xotirada 60 elementdan iborat bo'ladi, chunki har bir familiyagacha 0 lar bilan to'ldiriladi. Ko'rsatkichlar massivi yordamida bu massivni quyidagicha ta'riflash mumkin:

char *pf[]= {"Olimov", "Rahimov", "Ergashev"}.

Bu holda ro'yxat xotirada 23 elementdan iborat bo'ladi, chunki har bir familiya oxiriga 0 belgisi qo'yiladi. Ko'rsatkichlar massivlari murakkab elementlarni sodda usulda tartiblashga imkon beradi.

Ko'rsatkichlar massivlari funksiyalarda matritsalar qiymatlarini o'zgartirish uchun ishlatilishi mumkin. Quyidagi misolda matritsani transponirlash funksiyasi ishlatiladi:

```
#include <iostream.h> void trans(int n,double
*p[]) { double x;
for (int i=0;i<n—1;i++) for (int j=i+1;j<n;j++)
                                {x=p[i][j]; p[i][j]=p[]][i];
p[j][i]=x;
}
};
void main()
{double a[][3]={{11,12,13},{21,22,23},{31,32,33}};
double* ptr[3]={(double*)&a[0],(double*)&a[1], (double*)&a[2]};
int n=3;
trans(n,ptr);
for (int i=0;i<n;i++)
{cout<<"\n" <<i+1;
for (int j=0;j<n;j++)
cout<<" "<<a[i][j];
};
};
```

Dinamik massivlar

C++ tilida o'zgaruvchilar yo statik tarzda — kompilyatsiya paytida, yoki standart kutubxonadan funksiyalarni chaqirib olish yo'li bilan dinamik tarzda — dasturni bajarish paytida joylashtirilishi mumkin. Asosiy farq ushbu usullarni qo'llashda — ularning samaradorligi va moslashuvchanligida ko'rinadi. Statik

joylashtirish samaraliroq, chunki bunda xotirani ajratish dastur bajarilishidan oldin sodir bo'ladi. Biroq bu usulning moslashuvchanligi ancha past, chunki bunda biz joylash- tirilayotgan obyektning turi va o'lchamlarini avvaldan bilishimiz kerak bo'ladi. Masalan, matniy faylning ichidagisini satrlarning statik massivida joylashtirish qiyin: avvaldan uning o'lchamlarini bilish kerak bo'ladi. Noma'lum sonli elementlarni oldindan saqlash va ishlov berish kerak bo'lgan masalalar odatda xotiraning dinamik ajratilishini talab qiladi.

Xotirani dinamik va statik ajratish o'rtasidagi asosiy farqlar quyidagicha:

- statik obyektlar nomlangan o'zgaruvchilar bilan belgilanadi hamda ushbu obyektlar o'rtasidagi amallar to'g'ridan-to'g'ri, ularning nomlaridan foydalangan holda, amalga oshiriladi. Dinamik obyektlar o'z shaxsiy otlariga ega bo'lmaydi va ular ustidagi amallar bilvosita, ko'rsatkichlar yordamida, amalga oshiriladi;
- statik obyektlar uchun xotirani ajratish va bo'shatish kompilya- tor tomonidan avtomatik tarzda amalga oshiriladi. Dasturchi bu haqda o'zi qayg'urishi kerak emas. Statik obyektlar uchun xotirani ajratish va bo'shatish to'laligicha dasturchi zimmasiga yuklatiladi. Bu anchayin qiyin masala va uni yechishda xatoga yo'l qo'yish oson.

Dinamik tarzda ajratilayotgan xotira ustida turli xatti-harakatlarni amalga oshirish uchun new va delete operatorlari xizmat qiladi.

Shu paytga qadar barcha misollarda statik xotira ajratish qo'llanadi. Masalan, i o'zgaruvchisini aniqlash:

```
int i=1024;
```

Bu buyruq xotirada shunday sohani ajratib beradiki, u int turidagi o'zgaruvchini saqlash, ushbu soha bilan i nomini bog'lash hamda u yerga 1024 qiymatini joylashtirish uchun yetarli bo'ladi. Bularning hammasi dastur bajarilishidan oldin kompilyatsiya bosqichida amalga oshiriladi.

III. Dasturiy qism.

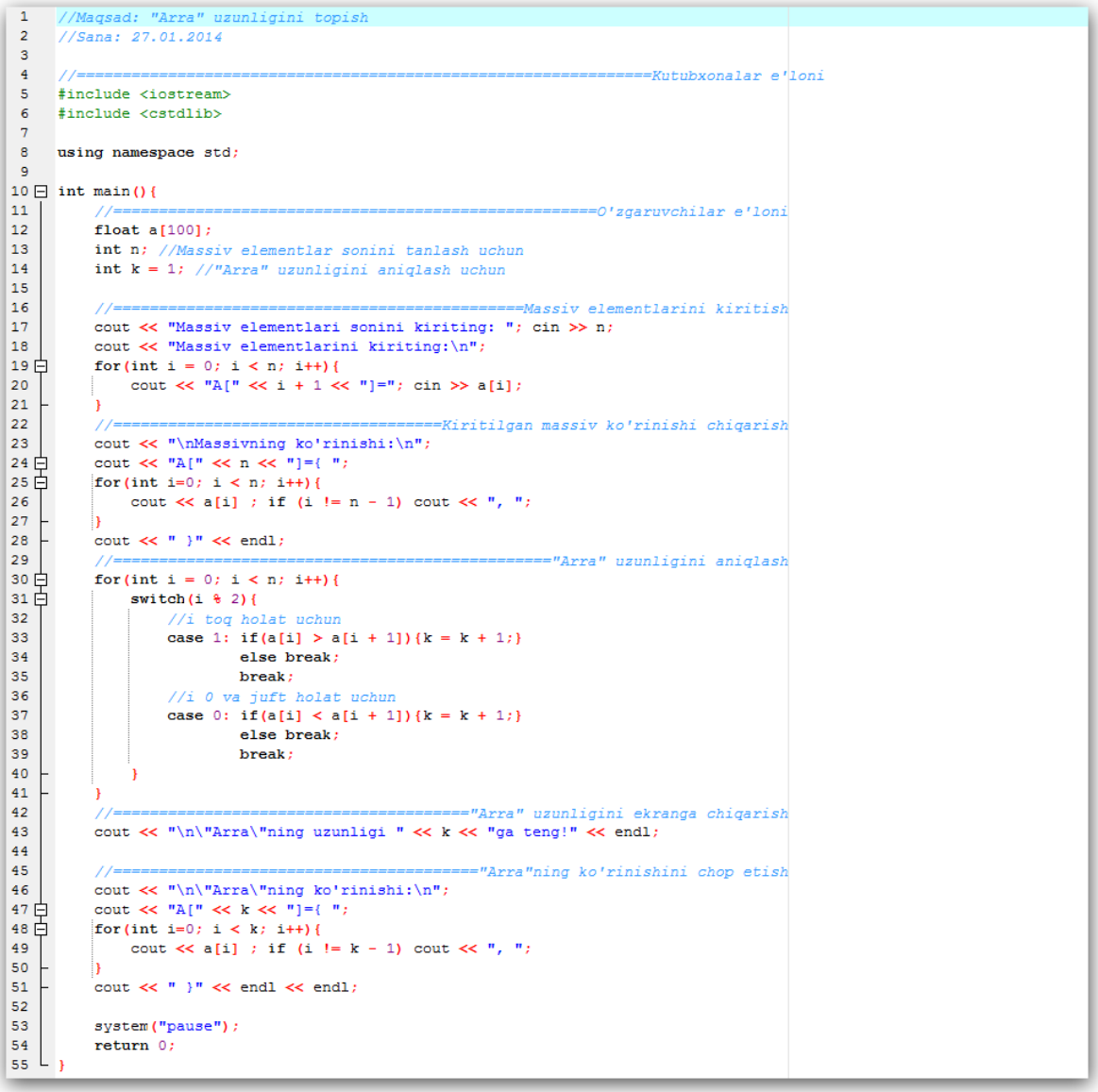
Dastur kodi:

```
//Maqsad: "Arra" uzunligini topish
//Sana: 27.01.2014
//==========================================Kutubxonalar e'loni
#include <iostream>
#include <cstdlib>
using namespace std;
int main(){
	//=====================================O'zgaruvchilar e'loni
	float a[100];
	int n; //Massiv elementlar sonini tanlash uchun
	int k = 1; //"Arra" uzunligini aniqlash uchun
	//================================Massiv elementlarini kiritish
	cout << "Massiv elementlari sonini kiriting: "; cin >> n;
	cout << "Massiv elementlarini kiriting:\n";
	for(int i = 0; i < n; i++){
		cout << "A[" << i + 1 << "]="; cin >> a[i];
	}
	//============================Kiritilgan massiv ko'rinishi chiqarish
	cout << "\nMassivning ko'rinishi:\n";
	cout << "A[" << n << "]={ ";
	for(int i=0; i < n; i++){
		cout << a[i] ; if (i != n - 1) cout << ", ";
```

```
}
cout << " }" << endl;
//====================================="Arra" uzunligini aniqlash
for(int i = 0; i < n; i++){
	switch(i % 2){
		//i toq holat uchun
		case 1: if(a[i] > a[i + 1]){k = k + 1;}
				else break;
				break;
		//i 0 va juft holat uchun
		case 0: if(a[i] < a[i + 1]){k = k + 1;}
				else break;
				break;
	}
}
//============================"Arra" uzunligini ekranga chiqarish
cout << "\n\"Arra\"ning uzunligi " << k << "ga teng!" << endl;
//==============================="Arra"ning ko'rinishini chop etish
cout << "\n\"Arra\"ning ko'rinishi:\n";
cout << "A[" << k << "]={ ";
for(int i=0; i < k; i++){
	cout << a[i] ; if (i != k - 1) cout << ", ";
}
cout << " }" << endl << endl;
```

```
    system("pause");

    return 0;

}
```

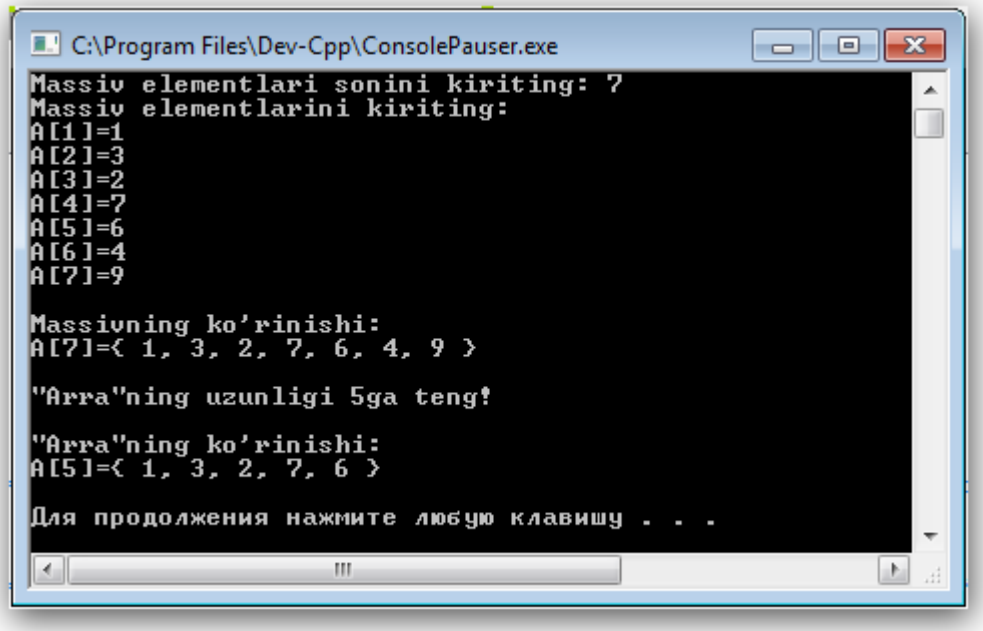
Dasturning Dev-C++dagi ko’rinishi:



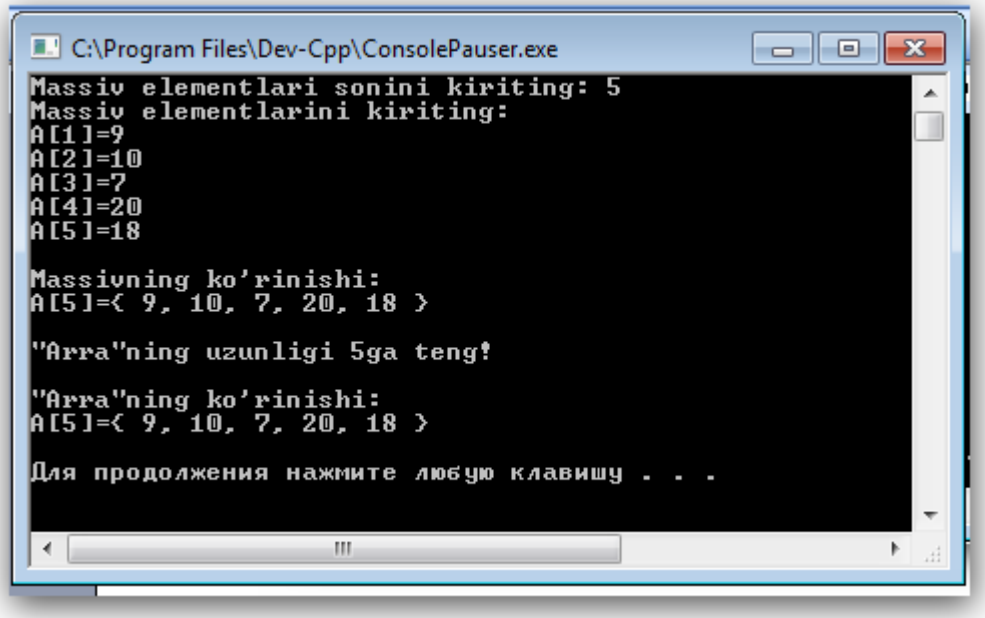
```
//Maqsad: "Arra" uzunligini topish
//Sana: 27.01.2014

//================================================================Kutubxonalar e'loni
#include <iostream>
#include <cstdlib>

using namespace std;

int main(){
    //=========================================================O'zgaruvchilar e'loni
    float a[100];
    int n; //Massiv elementlar sonini tanlash uchun
    int k = 1; //"Arra" uzunligini aniqlash uchun

    //=============================================Massiv elementlarini kiritish
    cout << "Massiv elementlari sonini kiriting: "; cin >> n;
    cout << "Massiv elementlarini kiriting:\n";
    for(int i = 0; i < n; i++){
        cout << "A[" << i + 1 << "]="; cin >> a[i];
    }
    //===================================Kiritilgan massiv ko'rinishi chiqarish
    cout << "\nMassivning ko'rinishi:\n";
    cout << "A[" << n << "]={ ";
    for(int i=0; i < n; i++){
        cout << a[i] ; if (i != n - 1) cout << ", ";
    }
    cout << " }" << endl;
    //=================================================="Arra" uzunligini aniqlash
    for(int i = 0; i < n; i++){
        switch(i % 2){
            //i toq holat uchun
            case 1: if(a[i] > a[i + 1]){k = k + 1;}
                    else break;
                    break;
            //i 0 va juft holat uchun
            case 0: if(a[i] < a[i + 1]){k = k + 1;}
                    else break;
                    break;
        }
    }
    //======================================"Arra" uzunligini ekranga chiqarish
    cout << "\n\"Arra\"ning uzunligi " << k << "ga teng!" << endl;

    //=========================================="Arra"ning ko'rinishini chop etish
    cout << "\n\"Arra\"ning ko'rinishi:\n";
    cout << "A[" << k << "]={ ";
    for(int i=0; i < k; i++){
        cout << a[i] ; if (i != k - 1) cout << ", ";
    }
    cout << " }" << endl << endl;

    system("pause");
    return 0;
}
```

Dasturning ishlash holatlariga misollar.

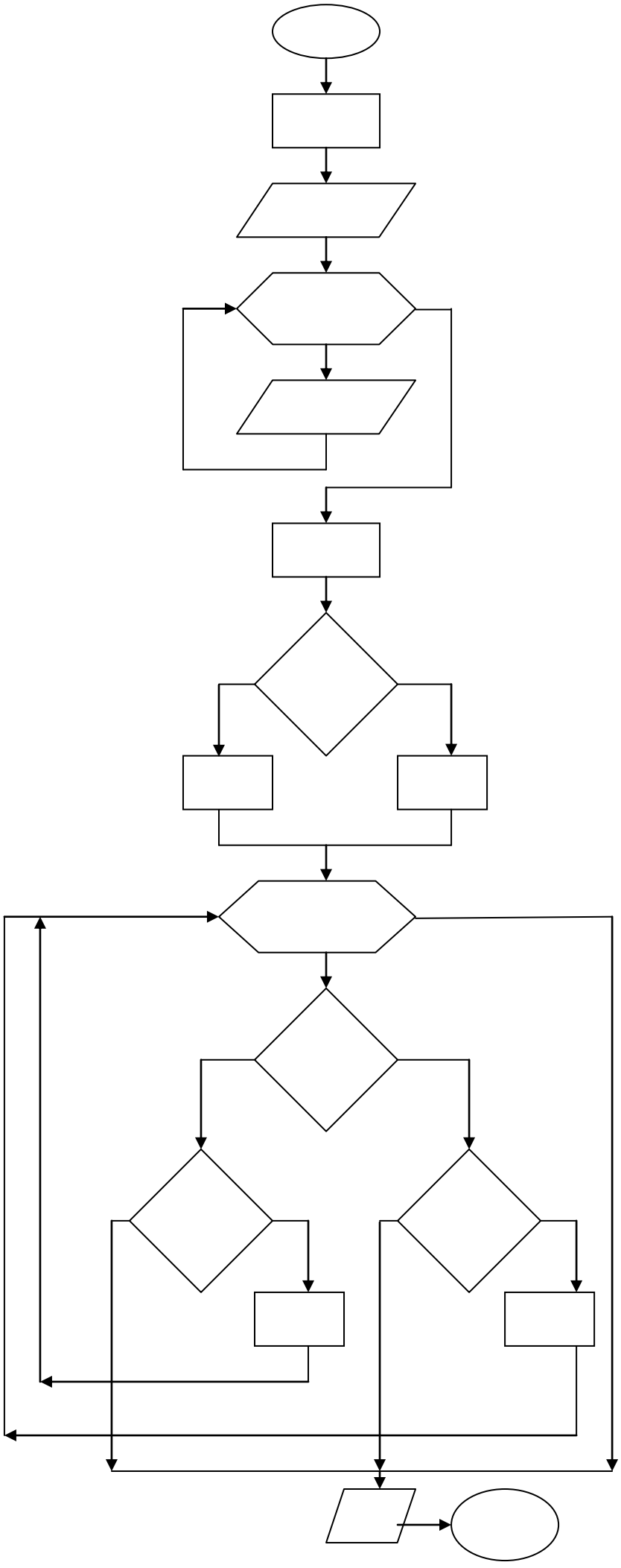
a)



b)



Dasturning blok sxemasi:



Xulosa

Men ushbu kurs ishimni bajarish davomida Dev-C++ dasturlash tilida o’rgangan ma’lumotlarim: o'zgaruvchilar, o’zgarmalar, identifikator, o'zgaruvchilarni e'lon qilish, matematik funksiyalar, dasturlash san'ati, mantiqiy amallar, takrorlash operatori, massivlar haqida olgan bilimlarimni yanada mustahkamladim. Bilimlarimni oshirib, yangi ma’lumotlatlarga ega bo’ldim. Men dastur tuzishda foydalangan operatorlarimni kengroq imkoniyatlarini bilib oldim. Dev C++ tilida amaliy dastur yaratish mobaynida shunga amin bo’ldimki, berilgan masalani yechish uchun birinchi navbatda uning mohiyatini to’liq anglab olish kerakligini tushindim. Chunki, masalani tushinib olish masalani yarim yechimi hisoblanishini bilamiz. Dastur tuzishda uning operatorlardan kerakli joyda kerakli tartibda foydalanish dasturning hajm jihatdan va sifat ko’rsatkichini yuqori darajada ekanligini ifodalaydi. Eng asosiysi dasturning foydalanuvchi bilan interfeysini yaxshi tashkillashimiz kerakligini o’rgandim. Men dasturimda foydalanganlarim jumladan massivlar haqida to’xtalib o’tsam. Massivlar ustida juda ko’plab masalalarni hal qilishimiz mumkin. Massivlarni juda mukammal ishlashni bilgan dasturchi har qanday dasturni hal qila olishiga amin bo’ldim.

Foydalanilgan adabiyotlar:

- Qudrat Abdurahimovning C++ dasturlash tilini o’rgatuvchi elektron qo’llanmasi.
- “C++ 21 день” kitobi
- Aslanov.K ”C++ dan qo’llanma.”
- Aripov.M ”Informatika. Informatika texnologiyalari” Toshkent 2004.
- Ayupov.L, Begalov.B, Ermatov.Sh, Shoahmedova.N “Shaxsiy kompyuterlar va samarali foydalanish asoslari”. O’quv qo’llanma Toshkent 2007.

Foydalanilgan saytlar:

- www.google.com
- www.dastur.uz
- www.bloodshed.net
- www.mingw.org
- www.youtube.com